CENTRUL DE PREGĂTIRE PENTRU PERFORMANŢĂ HAI LA OLIMPIADĂ !
DISCIPLINA INFORMATICĂ
JUDEŢUL SUCEAVA

Titlul lectiei: Algoritmul Fill
Data: 19.11.2016
Profesor: Ilincăi Florin
Grupa: clasa a X-a
Locul de desfasurare: CN PR Suceava

Problema 1

Algoritmul de umplere a unei suprafeţe închise. Se consideră o matrice binară. Valorile 1 delimitează o anumită suprafaţă închisă în cadrul matricei (elementele aparţinând acestei suprafeţe sunt marcate cu 0). Se dau coordonatele x şi y ale unui element al matricei, semnificând un punct din interiorul acestei suprafeţe. Se doreşte colorarea unei suprafeţe închise atunci când se cunoaşte coordonatele unui punct din interiorul ei.

Fie matricea:
0 1 1 0
0 0 0 1
0 1 1 1
1 0 0 0
Suprafaţa închisă este data de elementele A(1,1), A(2,1), A(2,2), A(2,3), A(3,1). Considerăm coordonstele (2,3) ale unui punct situat în interiorul acestei suprafeţe.
Matricea se bordează cu două linii şi două coloane cu valoarea 1. Aceasta are roul de a evita ieşirea din matrice.
Se utilizează funcţia ***scrie,*** o funcţie recursivă. Această funcţie funcţionează astfel:

- Testează dacă elemental matricei la care s-a ajuns, de coordonate (x, y) are valoarea 0.
- Dacă da, acesta ia valoarea 1, funcţia se autoapelează pentru fiecare dintre elementele învecinate: sus, jos, stânga, dreapta.
- În caz contrar se iese din funcţie.

```
#include <iostream>
#include <fstream>

using namespace std;
int a[100][100],i,j,m,n,x,y;
```

```
ifstream f("suprafata.in");
ofstream g("suprafata.out");
void scriu(int x, int y, int a[100][100])
{
   if(a[x][y]==0)
   {
      a[x][y]=1;
      scriu(x+1,y,a);
      scriu(x,y+1,a);
      scriu (x-1,y,a);
      scriu(x,y-1,a);
   }
}
int main()
{
  f>>n>>m;
  for(i=1;i<=m;i++)
   for(j=1;j<=n;j++)
       f>>a[i][j];
  for(i=1;i<=n;i++)
  {
     a[0][i]=1;
     a[m+1][i]=1;
  }
  for(i=1;i<=m;i++)
  {
    a[i][0]=1;
    a[i][n+1]=1;
  }
  cout<<"x="; cin>>x;
  cout<<"y="; cin>>y;
  for(i=1;i<=m;i++)
  {
     for(j=1;j<=n;j++)
      g<<a[i][j]<<" ";
      g<<'\n';
   }
   scriu(x,y,a);
   g<<'\n';
   for(i=1;i<=m;i++)
   {
      for(j=1;j<=n;j++)
       g<<a[i][j]<<" ";
       g<<'\n';
   }
   return 0;
```

}

Problema 2

Fiind data o matrice m x n, unde m,n apartin lui N si exista y elemente notate cu 1 si z elemente notate cu 0 iar y+z=m*n, aflati aria cea mai intinsa de elemente de 1.
Exemplu:
m=4, n=4
matricea:
0 1 1 0
0 1 1 0
1 0 0 0
0 0 1 1
Se va afisa: 4, adica regiunea care cuprinde elementele: (1,2),(1,3),(2,2),(2,3)

```
#include<fstream>
#include<iostream>

using namespace std;
int a[100][100],n,m;
int size;
void fill(int i,int j,int k)
{
   if(a[i][j]==1)
   {
      a[i][j]=k; // fiecarei element din regiune ii corespunde numarul zonei
      size++;
     /*verificam daca elementele alaturate pot apartine aceiasi regiuni si daca nu au fost
       inca marcate*/
      fill(i-1,j,k);
      fill(i+1,j,k);
      fill(i,j-1,k);
      fill(i,j+1,k);
   }
}

int main()
{
   int i,j;
   ifstream f("fill.in");
   f>>m>>n;
   for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
         f>>a[i][j];
   int max=0;
   int k=1;
   for(i=1;i<=m;i++)
```

```
    for(j=1;j<=n;j++)
        if(a[i][j]==1){
            k++; //fiecarei regiuni ii va corespunde un numar de ordine
            size=0;//numarul de elemente dintr-o regiune este initial 0
            fill(i,j,k);
            if(max<size) max=size;//calculam maximul ariilor
        }
  cout<<max<<endl;
  for(i=1;i<=m;i++)
  {
    for(j=1;j<=n;j++)
        cout<<a[i][j];
  cout<<endl;
  }return 0;
}
```

Problema 3

Problema fotografiei

O fotografie alb-negru este reprezentată sub foma unei matrici binare. Ea înfăţişează unul sau mai multe obiecte. Porţiunile corespunzătoare obiectului (sau obictelor) în matrice au valoarea 1. Se cere să se determine dacă fotografia reprezintă unul sau mai multe obiecte.
Fie matricea

1 1 0 0
0 0 0 1
1 1 1 1
1 1 1 1

Această fotografie conţine două obiecte.

Pentru a evita ieşirea din matrice, aceasta este bordată cu linii şi coloane având valoarea 0. Căutarea se va face pe 8 direcţii.

Se citeşte matricea şi se caută primul element 1 printre elementele acesteia. Se apelează funcţia compact care are rolul de a marca cu 0 toate elementele matricei care aparţin acestui prim obiect identificat. La revenire se testează dacă mai există valoarea 1 în matrice. În caz afirmativ se poate trage concluzia că în fotografie există mai multe obiecte, altfel fotografia conţine doar un obiect.

```
#include <iostream>
#include <fstream>

using namespace std;
int a[100][100],m,n,i,j,x,y,ok;
ifstream f("foto.in");
ofstream g("foto.out");

void compact(int x, int y, int a[100][100])
```

```
{
   if(a[x][y])
   {
      a[x][y]=0;
      compact(x-1,y,a);
      compact(x-1,y+1,a);
      compact(x,y+1,a);
      compact(x+1,y+1,a);
      compact(x+1,y,a);
      compact(x+1,y-1,a);
      compact(x,y-1,a);
      compact(x-1,y-1,a);
   }
}
int main()
{
   f>>n>>m;
   for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
      f>>a[i][j];
   for(i=1;i<=n;i++)
     {
      a[0][i]=0;
      a[m+1][i]=0;
     }
   for(i=1;i<=m;i++)
     {
      a[i][0]=0;
      a[i][n+1]=0;
     }
   x=0;
   do
   {
      x++;
      y=0;
      do
      {
         y++;
      } while(y!=n&&a[x][y]!=1);
   }while(x!=m&&a[x][y]!=1);
   compact(x,y,a);
   ok=0;
for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
      if(a[i][j]==1)
      ok=1;
```

```
if(ok)
   g<<"mai multe obiecte";
else
   g<<"un obiect";
   return 0;
}
```

Problema 4

Cerinţa
Se dă harta unui lac de formă dreptunghiulară, împărţit în *n*m* zone dispuse sub forma unei matrice cu *n* linii şi *m* coloane. Zonele pot fi acoperite cu apă, sau pot fi zone de uscat. Zonele de uscat care sunt învecinate pe linie sau pe coloană formează insule sau peninsule. Peninsule conţin cel puţin o zonă de uscat pe marginea lacului (matricei), în timp ce insulele sunt situate în întregime în interiorul lacului.
Cunoscând harta lacului, determinaţi numărul de insule şi numărul de peninsule.

Date de intrare
Fişierul de intrare *lac.in* conţine pe prima linie numerele *n m*. Următoarele *n* linii conţin câte *m* valori *0* sau *1*, separate prin câte un spaţiu. Valoarea *0* reprezintă o zonă acoperită cu apă, iar valoarea *1* reprezintă o zonă acoperită cu uscat.
Date de ieşire
Fişierul de ieşire *lac.out* va conţine pe prima două numerele *nrI nrP*, separate prin exact un spaţiu, reprezentând numărul de insule, respectiv numărul de peninsule existente pe lac.
Restricţii şi precizări

- *1 ≤ n , m ≤ 1000*

Exemplu
lac.in

```
6 9
1 1 0 0 1 1 1 0 0
0 0 1 0 1 0 1 0 0
1 1 0 0 0 0 0 0 0
1 0 0 0 1 1 1 0 0
1 1 1 1 0 1 0 0 0
1 0 0 0 0 0 1 1 0
```

lac.out

```
2 4
```

```
#include<fstream>
#include<iostream>

using namespace std;
ifstream f("lac.in");
ofstream g("lac.out");
int n,m,i,j,a[1000][1000],nri,nrp;
```

```
int ic,sc,coada [2][100000];
int dx[]={-1,0,1,0}; //elem din N este o linie în sus, aceeaşi coloană, elem din E, aceeaşi linie,
int dy[]={0,1,0,-1}; // o cloana la dreapta, S – o linie jos, aceeaşi col., V- aceeasi linie, col. stg

void fill(int x,int y)
{ int i;
ic=1;
sc=1;
 coada[0][1]=x;
 coada[1][1]=y;
 a[x][y]=0;
 while(ic<=sc)
    {
       for(i=0;i<=3;i++)
          if(a[coada[0][ic]+dx[i]][coada[1][ic]+dy[i]]==1)
       {
       sc++;
       coada[0][sc]=coada[0][ic]+dx[i];
       coada[1][sc]=coada[1][ic]+dy[i];
       a[coada[0][ic]+dx[i]][coada[1][ic]+dy[i]]=0;
       }
    ic++;
    }
}

int main()
{
f>>n>>m;
for(i=1;i<=n;i++)
    for(j=1;j<=m;j++)
          f>>a[i][j];
for(i=1;i<=n;i++)
    if(a[i][1]==1)
     {
        nrp++;
        fill(i,1);
     }
for(i=1;i<=n;i++)
    if(a[i][m]==1)
    {
       nrp++;
       fill(i,m);
    }
for(j=1;j<=m;j++)
    if(a[1][j]==1)
    {
```

```
        nrp++;
        fill(1,j);
    }
for(j=1;j<=m;j++)
    if(a[n][j]==1)
    {
        nrp++;
        fill(n,j);
    }
for(i=2;i<n;i++)
    for(j=1;j<m;j++)
    if(a[i][j]==1)
     {
         nri++;
         fill(i,j);
     }
g<<nri<<" "<<nrp;
    return 0;
}
```